Quadrilateral Mesh Smoothing

Xuan Huang

Advised by Professor Dianna Xu

February 18, 2018

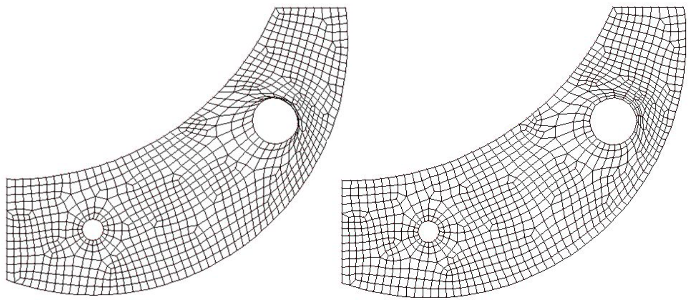


Figure 1: Quadrilateral meshes smoothing via vertex repositioning

1 Introduction

Mesh quality improvement is an important problem with a wide range of practical applications. Element quality of a mesh heavily affects the results of numerical simulation, such as rendering (Figure 2) or physical simulation (Figure 3) using that mesh. In the context of finite element mesh smoothing, vertex repositioning is the primary technique employed, where we allow vertex motion only and the connectivity of all edges remains unchanged. In the 2D cases we are interested in, this means that only the x y values of vertices can be modified. Maybe it would be nice to introduce triangular smoothing and quadrilateral smoothing here

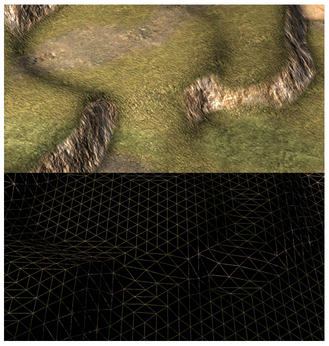
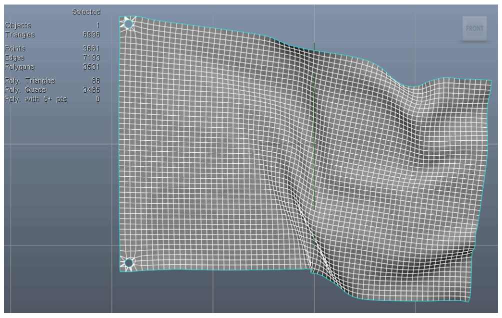


Figure 2: A example of terrain rendering



```
                 Selected
Objects                 1
Triangles            6986
Points               3661
Edges                7193
Polygons             3531
Poly. Triangles        66
Poly. Quads          3465
Poly. with 5+ pts       0
```

Figure 3: High-quality cloth simulation with NVIDIA

2 Element Quality

Element quality is measured either by max/min angles or aspect ratio (longest edge over shortest), or both. In a quadrilateral mesh the aspect ratio is defined as the largest ratio of longest edge over shortest edge among all elements, and max/min angles are the maximum/minimum among all angles. These two proprieties are related in triangles (by the law of sine), but the relationship is quite loose in quads (Figure 4). We investigate a smoothing method based on the idea of improving aspect ratio by circumscircles.

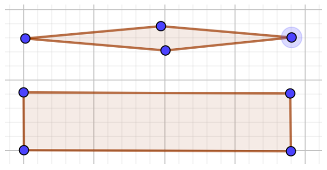


Figure 4: Aspect ratio and angles are not strictly related

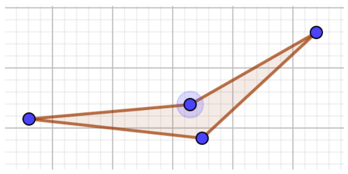


Figure 5: Non-convex quad

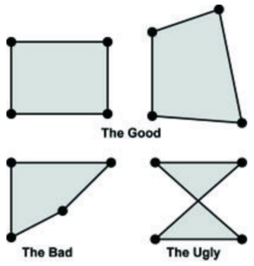


Figure 6: The good, the bad, and the ugly

Convexity is another important aspect of quadrilateral elements (Figure 5). It is very undesired to generate any element that has an angel greater than 180 degree. Again this is never a problem for triangles, and one of our goals will be trying to eliminate such cases from happening. Besides, self-intersection and edge flipping should be avoid under any circumstances. Figure 6 illustrates a general situation for desired and undesired quadrilateral elements.

Nice and clear!

“should be avoided”

3 Background

Existing methods usually divide into mesh modification via vertex insertion/deletion, edge/face swapping and remeshing [Dey and Ray 2010], or vertex repositioning without changing mesh connectivity [Amenta et al. 1997; Field 1988; Zhou and Shimada 2000]. Among those that do not modify mesh topology, Laplacian smoothing is the most commonly used because of its simplicity. In its most basic form, it moves each vertex to the centroid of the polygonal region formed by its neighboring vertices. It is a local method with very low computational cost, compared to the alternatives, which are optimization-based [Chen and Holst 2011; Freitag 1997; Parthasarathy and Kodiyalam 1991]. The most closely related work is Zhou and Shimadas angle-based Laplacian smoothing [Zhou and Shimada 2000], which is a variant of Laplacian smoothing.

4 Triangular Mesh Smoothing

4.1 Laplacian and its Angle Based Variation

The widely used Laplacian smoothing method is a computationally inexpensive, easy to implement local technique for triangular mesh smoothing. It simply moves each vertex to the centroid specified by the polygon formed by all it adjacent vertices [Amenta et al. 1997; Field 1988; Zhou and Shimada 2000]. The process is applied on every vertices of the mesh for several iterations, usually until the positions of all vertices converges or the mesh quality is not getting better.

Clear!

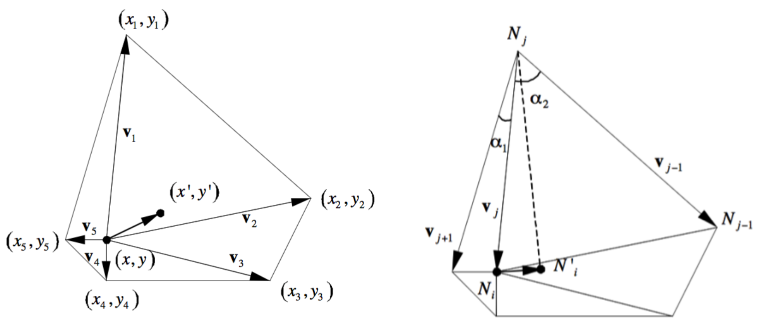


Figure 7: **Left**: The Laplacian method; **Right**: The angle-based variation. Rotating edge N_jN_i to angle bisectors to get the new position N_i' for vectex N_i

Problem arises for this method as it does not always move the vertices to optimal positions, and due to the fact that the centroid of a polygon could lie very close to, or even outside the shape itself, sometimes it generate invalid elements. Clear limitations of the related works

Zhou and Shimada presents an angle-based smoothing method based on the Laplacian smoothing algorithm. Instead of taking the centroid to be the new position of each vertex, they rotate each interior edge attached to the angle bisector, and take the average value resulted from all such repositioning actions to be the new position of the vertex. The result has a general improvements in mesh quality improving comparing to the original Laplacian method, and has largely reduced the possibility of getting invalid elements. It also provides a mechanism to implement the method on quad meshes simply by treating the quadrilateral element as the combination of two triangles.

4.2 Aspect-Ratio Based

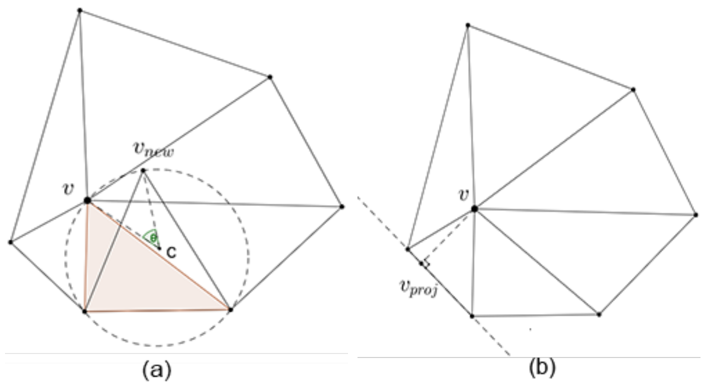


Figure 8: (a) Move v to v_{new}; (b) $\overline{vv_{proj}}$ is the shortest distance to star boundary

This section introduce our previous method to improve aspect ratio as an alternative to the existing angle-based method. We believe that aspect ratio will lead to a more balanced improvement in triangles, and this is the work motivated by aspect ratio improvements for quadrilateral meshes, which are unrelated to angles. Clear distinctions between the two approaches

What's a star region?

The idea is that inside the star region of each vertex, move the center vertex along the circumcircle of each incident triangle in a direction that will decrease the difference between each pair of adjacent interior edges. Each move must improve the aspect ratio of a triangle, because one of the two edges must be the shortest or longest edge of the face considered. Some restrictions are applied so that triangles already with good quality are not further modified and potentially sacrificed for worse ones.

The algorithm performs the following steps:

Description in steps is a good choice. Concise and clear!

1. As shown in Figure 8, for each vertex v not on the boundary there are k faces inside its star polygon. For each triangle if the two interior edges has a ratio higher than 1.5, calculate its circumcircle centered on C.
2. Move v clockwise or counter-clockwise towards the longer incident edge.
3. Let $\overline{vv_{proj}}$ be v's vertical distance to each star polygon edge. If the new position ends up decreasing the shortest distance from the vertex to the star polygon boundary, discard movement.
4. Take the mean of all positions after processing all k faces and assign the v there. Iterate until the resulting positions converges.

The max aspect ratio is improved over angle-based method in all cases tested. The algorithm might also provide some additional improvements on angles. It tends to recover seriously distorted areas and is also much less likely to result in invalid shapes compared to the angled- based method.

typeset

5 Current Plan

5.1 Challenges and goal

Goals?

We plan to develop a computationally inexpensive smoothing algorithm for quadrilateral meshes. As mentioned in the previous section, the triangular mesh smoothing methods do provide techniques via cutting the quads into two triangles and smooth the resulted triangular mesh. Apparently this way of smoothing ignores the proprieties special to quads, such as the loose angle - aspect ratio relationship (the measurement of quality could be a lot more complicated [Robinson 1987]), and the fact that a quad could become non-convex during the process. We would like our algorithm to be aware of such proprieties, and provide a more reliable solution to quadrilateral meshes specifically.

The goal is to first maintain the convexity, and then to provide a generally effective improvement on arbitrary input mesh. The process will be mostly experimental and the final result will be reflected on the outcome of the algorithm running on various examples. Likely the result from angle-based method will also be presented for comparison.

So the result mainly presented is the aspect-ratio-based method?

5.2 Our attempt

At this stage, we are still following the idea of splitting quads into two triangles and deal with the triangles first. The measuring factors remain to be only max/min angle and aspect ratios. The idea of utilizing circumscircle might be helpful in smoothing or measuring the quality of elements.

There are three major problems to consider at this point. First of all, We have to decide how to deal with the two triangles once the quad has been split. The original method we applied for triangular meshes moves each vertex on it circumscircle so that the aspect ratio of that single triangle will be improved. And since the circumscirlce also indicates the value of the angles, the other option is to try emerging the two circumscircle in someway, so that they become more similar to each other (if they overlap completely then we can possibly get a square by sliding vertices along the circumscircle).

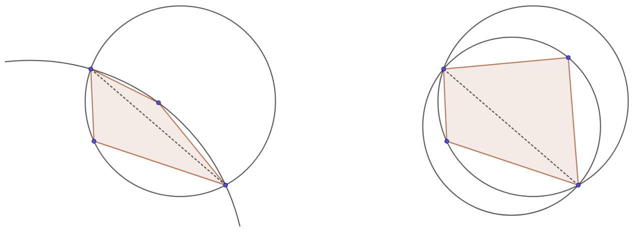


Figure 9: Two similar circumscircles imply better quad?

Secondly, we have to consider how to divide the quads into two triangles. Most of the time different divisions will result in different circumscircles, and we have to find a general rule of which way of splitting the quad is more desired. In the example given by Figure 11, the two individual triangles resulted from splitting could be both nice or both bad, for the exact same quad element.

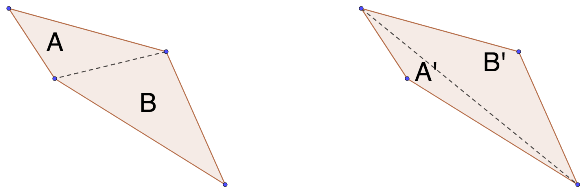


Figure 10: Different ways of splitting a quad

The order of smoothing also has to be determined. We can always sort the quads by their quality and start with all vertices on the worst one. Or we can go by vertex and determine the most urgent vertex to move at every step. Since we are doing thing locally, the priority has to be measured in some way that will guarantee a better result for the overall mesh.

potentially based on? What metrics?

Starting from a single quad, we are now experimenting ways of smoothing it. The current attempt is to correct the angles first, as the circumscircle is largely dependent on the value of angles and sliding vertices along circumscircle may eventually fix the aspect ratio. We decide to check each angle in the quad individually, if it is greater than a certain threshold (say greater than 120 degree), then the related triangle is likely too obtuse, and thus we push the vertex away from it opposite edge while shrinking the opposite edge itself. Similar approach is implemented for angles lower than a certain threshold (smaller than 30 degree).

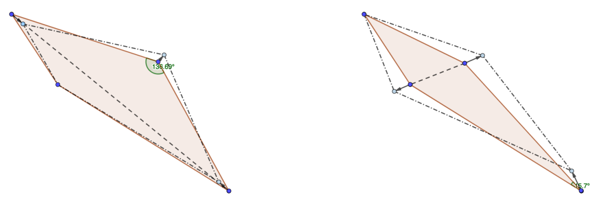


Figure 11: Smoothing scheme: **Left** Large angle; **Right** Small angle

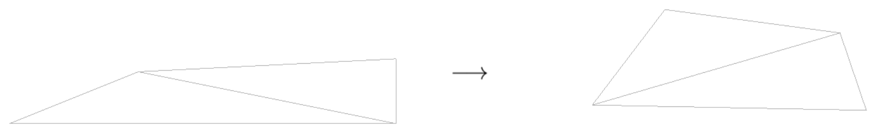


Figure 12: A set of test case

The result looks fine at this point as angles have a general tendency to improve after running 10-30 iterations. The aspect ratio is not taken care of under some circumstances, which is still okay at this point since we haven't paid attention to that yet. The next step is to put the method on a true quad mesh (instead of a single element), where edge flipping is to be expected in the worst case. To reduce such probability, the plan is to first employ the idea of averaging the result of several actions, and then to set a strict boundary test, as done in the aspect ratio based triangular mesh smoothing method.

Maybe explain this term?

The idea of using circumscircle to improve aspect ratio maybe incorporated at the same time. Depending on our observation from the test cases, this could be done alternatively with the angle smoothing process, or perhaps it is better to wait until the angles are good enough so that the circumscircle reflects the true shape of the entire quad instead of a single triangle.

6 Documentation

L. Chen and M. Holst. 2011. Efficient Mesh Optimization Schemes based on Optimal Delaunay Triangulations. Computer Methods in Applied Mechanics and Engineering 200 (2011), 967984.
T. Dey and T. Ray. 2010. Polygonal Surface Remeshing with Delaunay Refinement. Engineering with Computers 26 (2010), 298301.
D. Field. 1988. Laplacian Smoothing and Delaunay Triangulations. Communications in Applied Numerical Methods 4 (1988), 709712. Issue 6.
L. Freitag. 1997. On Combining Laplacian and Optimization-based Mesh Smoothing Techniques. AMD Trends in Unstructured Mesh Generation 220 (1997), 3743.
V. Parthasarathy and S. Kodiyalam. 1991. A Constrained Optimization Approach to Finite Elemement Mesh Smoothing. Finite Elements in Analysis and Design 9 (1991), 309320.
T. Zhou and K. Shimada. 2000. An Angle-based Approach to Two-dimensional Mesh Smoothing. In Proceedings, 9th International Meshing Roundtable. 373384.
J. Robinson. 1987 "CRE method of element testing and the Jacobian shape parameters", Engineering Computations, Vol. 4 Issue: 2, pp.113-118, https://doi.org/10.1108/eb023689

Figure 1. An example of quadrilateral mesh smoothing. Reprinted from Mesh Smoothing Challenges in the Industry, In Slide Player, N. Mukherjee, Retrieved February 17, 2018, from http://slideplayer.com/slide/5853324/. Copyright 2016 by N. Mukherjee.
Figure 2. An example of terrain rendering. Reprinted from Terrain texture projection artifacts, in Developer's blog for IceFall Games, Retrieved February 17, 2018, from http://cgicoffee.com/blog/2016/08/nvidia-flex-cloth-simulation, mtnphil. Copyright September 25, 2012 by mtnphil at Developer's blog for IceFall Games.
Figure 3. High-quality cloth simulation with NVIDIA. Reprinted from Maxwell Render/Learn/Render/Channel Rendering, in Next Limit, Retrieved February 17, 2018, from http://support.nextlimit.com/display/rf2016docs/Channel+Rendering. Copyright by RealFlow 10 Documentation.
Figure 4-5. Self made with GeoGebra at https://www.geogebra.org/.
Figure 6. The good the bad and the ugly. Reprinted from Plate/Shell in risa.com, Retrieved February 17, 2018, from https://risa.com/risahelp/risa3d/Content/3D_2D_Only_Topics/Plates.htm. Copyright by risa.com.
Figure 7 from T. Zhou and K. Shimada [2000].
Figure 8-12. Self made with GeoGebra at https://www.geogebra.org/ and Meshlab, modified with Adobe Photoshop.